IP Security Maintenance and Extensions
Internet-Draft
Updates: RFC7383 (if approved)
Intended status: Standards Track
Expires: 14 June 2026

A. Antony
S. Klassert
secunet
T. Brunner
codelabs GmbH

11 December 2025

IKEv2 Fragment Acknowledgment Extension
draft-antony-ipsecme-ikev2-fragment-acknowledgment-02

Abstract

This document specifies an extension to the Internet Key Exchange Protocol Version 2 (IKEv2) that enables acknowledgment of IKEv2 message fragments over UDP. The mechanism allows an IKE peer to confirm reception of individual fragments during the IKE_AUTH exchange and any subsequent exchanges where IKEv2 Fragmentation is used. Support for this feature is negotiated using a new Notify Message Status Type during IKE_SA_INIT, and fragment acknowledgments are exchanged using a separate Notification payload. This extension improves reliability when large IKE messages are exchanged, such as those containing post-quantum cryptography (PQC) payloads, and reduces retransmission overhead, thereby improving IKEv2 round-trip times in lossy network conditions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 June 2026.

Copyright Notice

Table of Contents

1. Introduction

The Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296] uses an unreliable transport (UDP) for message exchange.

Originally, IKEv2 messages were small — typically a few hundred bytes to a few kilobytes — such that a simple fragmentation [RFC7383] and retransmission mechanism operating over UDP, without congestion control or partial acknowledgments, was practically sufficient. However, with the introduction of post-quantum cryptographic (PQC) algorithms into IKEv2 [RFC9370], IKE peers are now required to exchange much larger messages than those produced by classical algorithms, often tens of kilobytes and sometimes approaching 64 kilobytes in size.

There are also several proposals to extend IKEv2 beyond the 64-kilobyte payload limitation [I-D.nir-ipsecme-big-payload], [I-D.smyslov-ipsecme-ikev2-extended-pld], [I-D.tjhai-ikev2-beyond-64k-limit].

In the current IKEv2 fragmentation mechanism [RFC7383], when one or more fragments are lost, the sender retransmits all fragments of the message. Practical experience shows that this can lead to significant retransmission overhead and long delays when large fragmented messages are exchanged. In some chronic cases, peers may fail to establish an IKE SA even after dozens of retransmissions. This document proposes a fragment acknowledgment mechanism for IKEv2, similar in concept to acknowledgment schemes used in QUIC [RFC9000].

When both the responder and initiator support the new IKEv2 Fragment Acknowledgment, the initiator retransmits only the fragments that the responder reports as missing, reducing bandwidth consumption and latency overhead.

The current IKEv2 retransmission model is entirely initiator-driven: only the initiator can decide when to retransmit a message after a timeout [RFC7383]. The responder has no means to request retransmission or to signal that it has received an incomplete set of fragments. This document proposes to extend that model slightly by allowing the responder, upon receiving one or more fragments of an

IKE message and detecting that some fragments are missing, to send an IKEv2 response, of the same exchange with Fragment Acknowledgment notification indicating the missing fragments. This message is sent with the IKEv2 Response flag set. It does not require a response, does not advance the IKEv2 Message ID state.

1.1. Terminology

This document uses the following terms defined in [RFC7296]: IKE_SA_INIT, IKE_AUTH, CREATE_CHILD_SA, SK_e, SK_a.

This document also uses the following terms defined in [RFC9242]: IKE_INTERMEDIATE.

This document also uses the following terms defined in [RFC7383]: IKEv2 Fragmentation, Total Fragments,

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Fragment Acknowledgment Notify Message Status Type payload

Two Notify Message Status Types are defined: FRAGMENT_ACK_REQ, which acknowledges fragments of an IKE request, and FRAGMENT_ACK_RES, which acknowledges fragments of an IKE response.

The Fragment Acknowledgment notifiers (FRAGMENT_ACK_REQ and FRAGMENT_ACK_RES) are primarily useful during the IKE_AUTH, IKE_INTERMEDIATE, and CREATE_CHILD_SA exchanges. During IKE_AUTH and IKE_INTERMEDIATE, peer authentication may still be incomplete, but the fragments themselves are protected by encryption and integrity using SK_e and SK_a, which have been derived but not yet authenticated. The format of the fragment header is specified in [RFC7383].

Using two distinct notifiers makes the direction of acknowledgment explicit, avoiding ambiguity in cases of simultaneous exchanges or delayed delivery where multiple messages share the same Message ID.

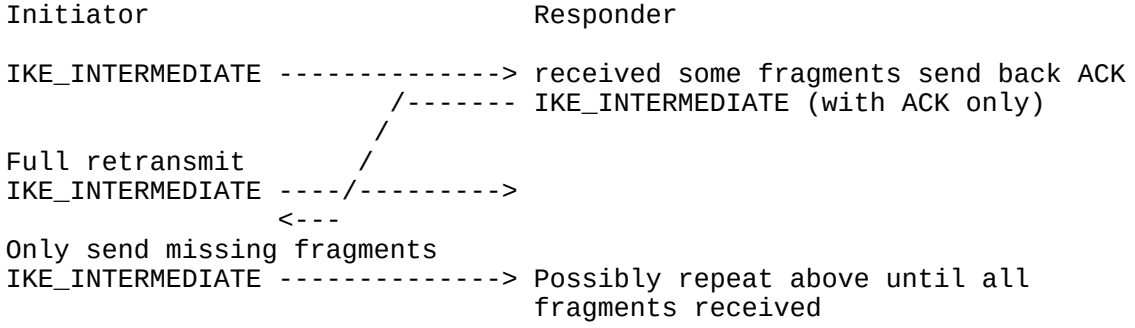

Figure 1: IKE_INTERMEDIATE request

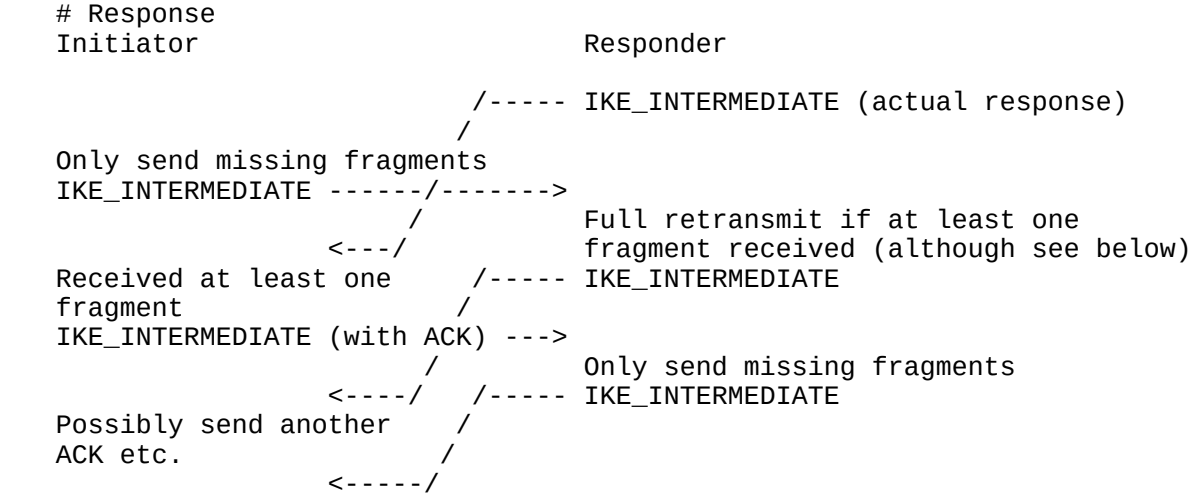

Figure 2: IKE_INTERMEDIATE response

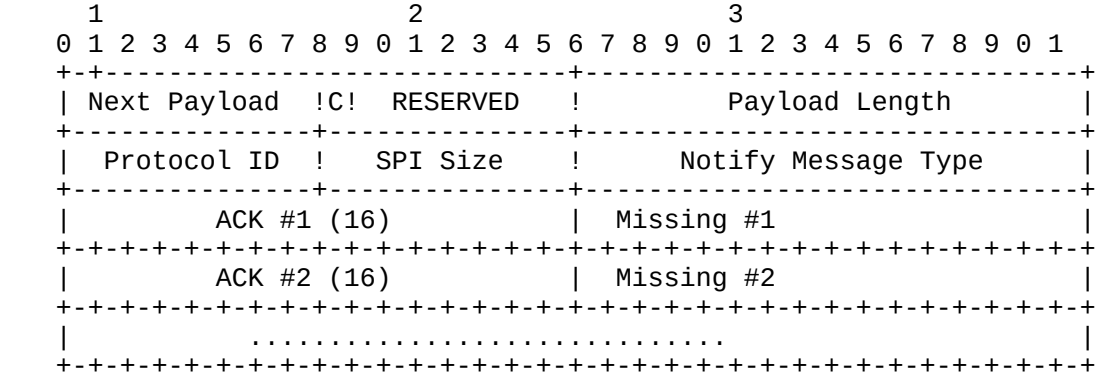

Figure 3: FRAGMENT_ACK

* Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.

* SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0.

* Notify Status Message Type value (2 octets) - set to TBD2 or TBD3

* Pairs of ACK # and Range

The payload enumerates the set of missing fragments for a single IKE message. The ACK # field identifies the lowest-numbered missing fragment, and the Missing # field specifies the contiguous range of additional missing fragments. This allows the sender to selectively

retransmit only those fragments that have not been received. The Total Fragments field is present in each fragment header, as defined in [RFC7383], Section 2.5. For example: 4,4; 7,10; and so on.

FRFRAGMENT_ACK_RESAGMENT_ACK_REQ or FRAGMENT_ACK_RES depends on whether a peer is acknowledging an IKEv2 request or a response.

3.1. Sending Fragment Acknowledgment response

The responder MAY send a response to an exchange with Fragment Acknowledgment notification after receiving one or more fragments of a request. See Figure 1. Similarly, the initiator MAY send a Fragment Acknowledgment notification after receiving one or more fragments of a response. See Figure 2.

When the initiator sends a FRAGMENT_ACK_RES notification in response to a fragmented response, it MUST set the IKE header Response (R) bit. This results in a message that is technically a “response to a response” and reuses the same Message ID, which is unusual in IKEv2. However, the notifier is distinct FRAGMENT_ACK_RES. This message is informational in nature, does not advance the Message ID state, and does not require a response.

3.2. Processing Fragment Acknowledgment Message

Unlike typical IKEv2 exchanges, which complete when a response with the matching Message ID arrives, Fragment Acknowledgment notification do not indicate completion of the exchange. Instead, this message requests retransmission of the missing fragments and MUST NOT advance the IKEv2 Message ID counter.

When the sender retransmits in response to a Fragment Acknowledgment, it SHOULD begin with the lowest missing fragment. (See editor’s note below regarding potential use of INFORMATIONAL exchanges.)

4. Negotiation

The use of Fragment Acknowledgment MUST be negotiated during the IKE_SA_INIT exchange. Both the initiator and the responder indicate support for this extension by including the FRAGMENT_ACK_SUPPORTED Notify Message Status Type (value TBD1) in the IKE_SA_INIT request and response messages. The presence of this notification in both directions confirms that both peers support the Fragment Acknowledgment mechanism. In addition, both peers MUST also signal support for IKEv2 Fragmentation by sending the IKEV2_FRAGMENTATION_SUPPORTED notification as specified in Section 2.3 Negotiation of [RFC7383].

If either peer omits the FRAGMENT_ACK_SUPPORTED notification in IKE_SA_INIT the extension MUST NOT be used in subsequent exchanges within that IKE SA.

The FRAGMENT_ACK_SUPPORTED notification follows the general rules for Notify Message Status Types as specified in [RFC7296], Section 3.10. It does not include any data in the Notification Data field.

5. Backward Compatibility

Receipt of a FRAGMENT_ACK_REQ or FRAGMENT_ACK_RES notification MUST NOT be interpreted as advancing the IKEv2 exchange state. Instead it is signal to retransmit only the missing fragments. It MUST only be sent after both peers have indicated support for FRAGMENT_ACK_SUPPORTED.

6. Operational Considerations

The FRAGMENT_ACK notification message SHOULD NOT be large enough to cause path-MTU issues. If the number of acknowledged fragments results in a payload that approaches the path MTU or the IKEv2 fragment size, the sender MAY limit the number of missing fragment ranges included in a single message and send multiple FRAGMENT_ACK messages if necessary. Implementations SHOULD ensure that each FRAGMENT_ACK message fits within a single UDP datagram to avoid IP-layer fragmentation.

When the sender receives multiple FRAGMENT_ACK messages, the sender treats the union of all ranges acknowledged so far as the authoritative view.

6.1. When to Send a Fragment Acknowledgment

Since the IKEv2 retransmission model is sender-driven, the responder SHOULD be conservative about when to send the first FRAGMENT_ACK message. A responder MAY send a FRAGMENT_ACK only after it detects that the sender has retransmitted at least one fragment of the same message, indicating that one or more fragments were likely lost. In general, an IKE peer sending a FRAGMENT_ACK message SHOULD do so only when it can provide useful information about missing fragments, in order to avoid unnecessary traffic and message processing overhead and to remain consistent with the sender-driven retransmission model of IKEv2. However, with large number of fragments this may take long time to converge.

6.2. IV reuse when using AEAD

One potential implementation issue I can see with these ACKs is the IV when using AEAD. Both the request and the response use the same Message ID as the actual messages more than once. If the IKEv2 Message ID is used as IV this would lead to reuse of IV. Which MUST be avoided.

6.3. Update to RFC7383

[RFC7383] specifies that when an IKE message is retransmitted, all fragments of that message, including the first fragment, MUST be retransmitted. The selective retransmission enabled by the FRAGMENT_ACK mechanism defined in this document relaxes that requirement by allowing the sender to retransmit only the fragments that the peer identifies as missing. Accordingly, this document updates [RFC7383] by modifying the retransmission behavior specified in Section 2.6. When FRAGMENT_ACK is negotiated, a sender MAY retransmit only the fragments indicated as missing by the peer, rather than retransmitting all fragments including fragment number 1.

Section 4 of [RFC7383] should be updated to allow fragment acknowledgment.

7. Editors Notes Open Issues

The following issues are retained for discussion during the evolution of this document and are expected to be resolved or removed if the document becomes a Working Group item.

* Path MTU discovery mentioned in [RFC7383] is currently ignored; applicability TBD.

* When fragments exceed the path MTU, they may not be acknowledged, and the IKE state will not advance. This will have re-fragmented as specified in [RFC7383]

* Why not use IKEv2 INFORMATIONAL? That may adhere more to IKEv2. However, every INFORMATIONAL needs a response. And if there is no response the INFORMATIONAL message MUST be retransmitted to advance IKE state. This would lead to complex unpredictable retransmissions.

* Why not make new IKEv2 exchange without a response? Instead of responding to the same : responding with IKE_AUTH or IKE_INTERMEDIATE This is worth considering. New Exchange IKE_FRAG_ACK : which has no response. The message will carry IKE exchange and message ID it is responding to. This might be bigger change. This will be a bigger protocol change. Probably too complicated?

* Should every acknowledgment sent by an IKE peer MUST include the entire state of the buffer. That is, acknowledgments are cumulative. However, when the entire state does not conservatively fit into one packet FRAGMENT_ACK message would be partial.

* Rename ACK to NACK? -ve acknowledgment? or SACK May be later?

7.1. New Exchange: IKE_FRAG_ACK?

A possible design alternative is to define a new IKEv2 exchange type, IKE_FRAG_ACK, which carries fragment acknowledgment information but does not have a response. Each IKE_FRAG_ACK message would include the Exchange Type and Message ID of the IKE message it acknowledges. This exchange has no response specified. It is one shot message. This approach would decouple fragment acknowledgment from existing IKE exchanges such as IKE_AUTH, IKE_INTERMEDIATE or CREATE_CHILD_SA.

7.2. Why not TCP?

Reliable transport for IKEv2 over TCP, as proposed in [I-D.ietf-ipsecme-ikev2-reliable-transport], adds implementation complexity and resource cost. It requires maintaining both TCP and UDP sockets, increasing energy use on low-powered devices. Using TCP for IKE while keeping ESP in UDP mode through NAT gateways introduces additional state and resource requirements. It may also be less compatible with hardware offloading and inefficient for low-power or mobile platforms.

Antony’s position is that using TCP for IKEv2 is not an ideal solution for improving reliability. While a UDP based ideas borrowed from QUIC-based could provide reliable transport and may be one day congestion control! It would be complex for the limited goal of fragment acknowledgment and selective retransmission. Other authors may have different views on this topic.

8. IANA Considerations

This document defines one new registration for the IANA "IKEv2 Notify Message Status Types" registry.

Value	Notify Message Status Type	Reference
[TBD1]	FRAGMENT_ACK_SUPPORTED	[this document]
[TBD2]	FRAGMENT_ACK_REQ	[this document]
[TBD2]	FRAGMENT_ACK_RES	[this document]

Table 1

9. Acknowledgments

ACKs TBD

10. Security Considerations

TBD

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <https://www.rfc-editor.org/info/rfc7383>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12. Informative References

[I-D.ietf-ipsecme-ikev2-reliable-transport]
Smyslov, V. and T. Reddy.K, "Separate Transports for IKE and ESP", Work in Progress, Internet-Draft, draft-ietf-ipsecme-ikev2-reliable-transport-00, 6 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-reliable-transport-00>.

[I-D.nir-ipsecme-big-payload]
Nir, Y., "A Larger Internet Key Exchange version 2 (IKEv2) Payload", Work in Progress, Internet-Draft, draft-nir-ipsecme-big-payload-06, 14 September 2025, <https://datatracker.ietf.org/doc/html/draft-nir-ipsecme-big-payload-06>.

[I-D.smyslov-ipsecme-ikev2-extended-pld]
Smyslov, V., "Extended IKEv2 Payload Format", Work in Progress, Internet-Draft, draft-smyslov-ipsecme-ikev2-extended-pld-01, 6 March 2023, <https://datatracker.ietf.org/doc/html/draft-smyslov-ipsecme-ikev2-extended-pld-01>.

[I-D.tjhai-ikev2-beyond-64k-limit]
Tjhai, C., Heider, T., and V. Smyslov, "Beyond 64KB Limit of IKEv2 Payloads", Work in Progress, Internet-Draft, draft-tjhai-ikev2-beyond-64k-limit-03, 28 July 2022, <https://datatracker.ietf.org/doc/html/draft-tjhai-ikev2-beyond-64k-limit-03>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9242] Smyslov, V., "Intermediate Exchange in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 9242, DOI 10.17487/RFC9242, May 2022, <https://www.rfc-editor.org/info/rfc9242>.

[RFC9370] Tjhai, CJ., Tomlinson, M., Bartlett, G., Fluhrer, S., Van Geest, D., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 9370, DOI 10.17487/RFC9370, May 2023, <https://www.rfc-editor.org/info/rfc9370>.

Appendix A. Additional Stuff

TBD

Authors' Addresses

Antony Antony
secunet Security Networks AG
Email: antony.antony@secunet.com

Steffen Klassert
secunet Security Networks AG
Email: steffen.klassert@secunet.com

Tobias Brunner
codelabs GmbH
Email: tobias@codelabs.ch